TECHNICAL REPORT 3048
SEPTEMBER 2016

Learning and Visualizing Modulation Discriminative Radio Signal Features

Michael Walton
Daniel Gebhardt, Ph.D.
Benjamin Migliori, Ph.D.
Logan Straatemeier

SSC Pacific
San Diego, CA 92152-5001

SSC Pacific
San Diego, California 92152-5001

K. J. Rothenhaus, CAPT, USN
Commanding Officer

C. A. Keeney
Executive Director

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the IO Support to National Security Branch (Code 56120), the IO Spectrum Exploitation Branch (Code 56140), and the Advanced IO Operations Branch (Code 56150) of the Information Operations Division (Code 56100), Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA. The SSC Pacific Naval Innovative Science and Engineering (NISE) Program funded this Basic Research project.

Released under authority of
G. Settelmayer, Head
Information Operations Division

SB

EXECUTIVE SUMMARY

In this work we explore the adaptation of convolutional autoencoders to complex-valued temporal domain radio signals. We propose a method for accomplishing online semi-supervised learning with a tied-weight convolutional autoencoder applied to a modulation classification task and provide some initial results. We also demonstrate a novel application of class activation maps (CAMs) [1] to obtain interpretable visualizations of modulation-discriminative temporal structure in input signals. Finally, we show that our visualization method may be successfully applied to pre-trained models with negligible impact on classification performance on an automated modulation classification (AMC) task.
This work was done as part of the BIAS (Biologically Inspired Autonomous Sensing) project, funded from the Naval Innovative Science and Engineering (NISE) Program.

CONTENTS

Figures

Tables

1. INTRODUCTION

The resurgence of deep neural networks has produced state-of-the-art advances in computer vision and an increasing number of other problem domains. One of the central advantages of these models is their ability to efficiently extract domain-specific features jointly when learning a predictive model. A still outstanding limitation, however, is a comprehensive means of interpreting the representations and latent statistical relationships learned by deep networks. In this work, we explore a particular class of models *Convolutional Autoencoders* for unsupervised and semi-supervised feature learning. In the later sections of this report, we also discuss a technique, *Class Activation Maps*, which compute the class significance of input regions for the features learned by a neural network. This method provides a means of interpreting the latent representations the network uses to make its predictions. We argue that this is particularly important in the RF signal processing domain where the application of deep neural networks is still in its infancy.

1.1 AUTOENCODERS

In this section, we review the general definition of autoencoders, summarize a few key concepts, and cast them in the context of modern deep learning. An autoencoder is an unsupervised neural-network model that learns an encoding of inputs for feature extraction. In its most general form, an autoencoder is composed of an encoder network ϕ that maps inputs X to a latent space Z, and a decoder network ψ which attempts to reconstruct the input. Commonly, the objective function to be minimized by this class of models is simply the squared error between the input and its reconstruction, which is often referred to as the 'reconstruction loss' $L_{rec} = ||X - \psi \circ \phi(X)||_2$, where $||x||_p$ is the p-norm of a vector x.

A degenerate or otherwise unuseful case of an autoencoder is one that simply learns the identity function. Introducing a "bottleneck" into the model enforces non-identity solutions as well as learn-ing a compressed representation of the data. If $X \in \mathbb{R}^n$ and $Z \in \mathbb{R}^m$ and $m < n$, the encoder network is a dimensionality reduction of X. Additionally, if ϕ is parameterized by a weight matrix W and $\psi = W^T$, this method is known as weight tying and has the desirable properties of reducing the parameter count and preventing disproportionate weights in the encoder and decoder pathways. Spe-cial cases of tied weight autoencoders also have interesting relationships to other unsupervised meth-ods. For instance, a linear model parameterized by the $n \times m$ weight matrix W trained to minimize L_{rec} posses a unique global optimum corresponding to the first m principal components (eigenvectors of the covariance matrix) associated with training exemplars [2].

Imposing sparse structure on Z prevents the model from learning the identity as well as improving generalization and robustness to noise for various problems [3] [4]. This may be accomplished by imposing sparsity regularizations on Z by stochastically setting its elements to zero during training (also known as dropout [5]) or a adding a sparsity term to the loss function, such as $||Z||_1$.

A stacked autoencoder is a model in which multiple, hierarchical latent representations are formed by training each layer to reconstruct the output of the previous layer. In this paradigm, layer-wise training is conducted by training the first layer $\ell_0 : X \rightarrow Z_0$ and $\ell_0^T : Z_0 \rightarrow X$ [1] using the back-propigation algorithm. Subsequent layers are then trained $\ell_k : Z_{k-1} \rightarrow Z_k$, where Z_k is the k^{th} latent representation.

[1] Here we assume tied-weights for simplicity; in subsequent definitions this is implied unless otherwise stated.

1.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) have been broadly applied to numerous problems in computer vision and natural language processing. Generally, a CNN learns a hierarchically structured decomposition of its input space as a set of convolutional filters. In supervised learning, as is commonly the task of CNNs, the model may jointly learn a probabilistic mapping of these features to a set of semantic concepts.

We will use the terms *instance* or *example* to refer to a particular input $x \in X$. A *sample* may refer to either a randomly drawn subset $\hat{X} \subseteq X$ or a single measurement x_t in a particular instance and should be clear from context. Because much of the work in CNNs has been conducted in the computer vision domain, much of the terminology and notation assumes $c \times w \times h$ input where c is the color channel or feature dimension[2] and w, h are the two dimensional spatial resolution of the input image. In the temporal radio-frequency domain, our signals are $c \times s$ where c is the number of channels and s is the number of samples per instance. Unless otherwise indicated, we define all operations (convolutions, downsampling, normalizations etc.) to operate across the last dimensions of the input signals, treating the c dimension as independent, which implies spatial and sample-wise operations in the image and RF domains respectively.

When discussing convolutions, we will use the terms feature, filter, and kernel more or less interchangeably. In the RF domain, we assume all kernels are $1 \times k$ with a stride of s and padding p; for images, we will often assume square inputs, square kernels, equivalent padding, and equivalent strides across both axes. However, note that the definitions and many of the properties of the operations discussed generalize to non-square and N-dimensional cases [6].

1.3 CONVOLUTIONAL AUTOENCODERS

A convolutional autoencoder (CAE) is simply an autoencoder that incorporates model design principals from CNNs. Models of this form have numerous applications, including deep-feature visualization [7] dense prediction [3] [8], generative modeling [9] [10], and semi-supervised learning [11]. Additionally, our recent work has shown that CAEs may also be used for blind signal denoising [4].

The encoder of a CAE is a standard CNN; the structure of the decoder, however, is non-trivial. In classical autoencoders, where the encoder is simply a succession of matrix-vector products interleaved with non-linear activation functions, an appropriate definition for the decoder follows naturally by adopting a "mirrored" structure. That is, the weights of each decoder layer learn to estimate the expected preimage of the corresponding transform in the encoder. However, the constraints imposed on convolutional network models leave the inverting structure of many common operations poorly defined. Many inverse operations have been proposed for common CNN layers, in particular, convolutions and max pooling. Here we will highlight some of these emerging methods and discuss their advantages and limitations as well as provide motivation for the particular combination of methods we have found most useful.

[2] If the input to a particular operation is X, this is often referred to as a "channel"; if instead, the input to an operation is the output of some mapping, this may be referred to as a feature.

[3] Also known as semantic segmentation in which every element of the input is assigned to a semantic category; for Instance, labelling all the pixels that comprise a cat.

[4] This work is documented in a technical report authored by Dr. Ben Migliori and is currently under review.

1.3.1 Inverting Max Pooling

Pooling operations subsample a signal to produce a single output from subregions of the input. In images, this is commonly implemented as non-overlapping rectangular blocks that are integrated by taking the mean (mean pooling) or max (max pooling) of each region. Of the two methods, max pooling is far more efficient to compute and more commonly used; for concision we only consider the latter method. Naive approaches to reversing the max pooling operation by interpolation have been explored extensively in [8] [11]. Advantages of this method are straightforwardness of implementation, and the ability to perform the up-sampling without additional side information (as we will explore in later methods). However, these interpolations can be costly to compute, as well as introducing dilations of the signal, which make it difficult for the model to reconstruct high-frequency details and sharp transitions.

An alternative upsampling method is proposed in [12], which is easy to compute and reduces the degree to which the signal is blurred. Given a feature map $\mathcal{F}$ that has been pooled by some factor (n, m), a sparse upsampling is computed as the kroneker product $\mathcal{F} \otimes M$ with a block matrix "mask" $M \in \{0, 1\}^{n \times m}$ with a single non-zero entry (by convention this element is the upper left corner of the mask for two dimensional inputs). This method yields impressively sharp reconstructions for shal-low models and can estimate the pre-image of arbitrary image feature transforms (CNN, SIFT, HOG, etc.) without additional knowledge of their structure or mechanism [13]. However, it is discussed in [12] that this method is discarding available information from the encoder in the case of convolutional autoencoders. Further, Dosovitskiy observes that their models' ability to reconstruct drops rapidly as a function of model depth.

A natural extension to max pooling is to define an operation that returns the $argmax$ of each pooling region, that is, the i, j coordinates of each maxima of a feature in the encoder in response to a particular input. To invert the max pool, it is then straightforward to construct a representation of the same dimension as the input with the maxima inserted at coordinates i, j and zeros elsewhere. This approximate inverse of the pooling operation, commonly referred to as "unpooling" [7], produces a sparse upsampling as in [12] without discarding structural information by arbitrarily inserting the maxima. In the literature, the coordinates of the maxima are commonly referred to as *switches* or *switch variables*, and for consistency we will use these terms as well.

1.3.2 Inverting Convolutions

The notion of reversing convolutional neural networks was first introduced in [14] and termed a *projection operation* by Matthew Zeiler in [7]. There has been much debate in recent literature as to the appropriate name for Zeiler's projection operation. Among these the terms *deconvolution, transpose convolution, upconvolution, fractionally strided convolution* and *backward strided convolution* have all been suggested. While these terms have varying degrees of clarity and validity, we adopt the term *transpose convolution* for reasons that will become evident from our discussion.

Note that the filters used in the decoder may have their own parameters, as in [11], or share weights with the corresponding filter in the encoder (harkening back to the notion of weight-tying in traditional autoencoders), as in [14] [7]. For the simplicity of the current discussion, we assume that the transpose convolution has some complementary encoding convolution (tied weights), as we will observe this need not be the case.

Transpose convolutions can be understood from several different perspectives, all of which help to justify the approach and clarify the mechanism. First, we consider transpose convolution as an **interpolating filter**. The unpooling operation produces a sparse activation map with the same dimension

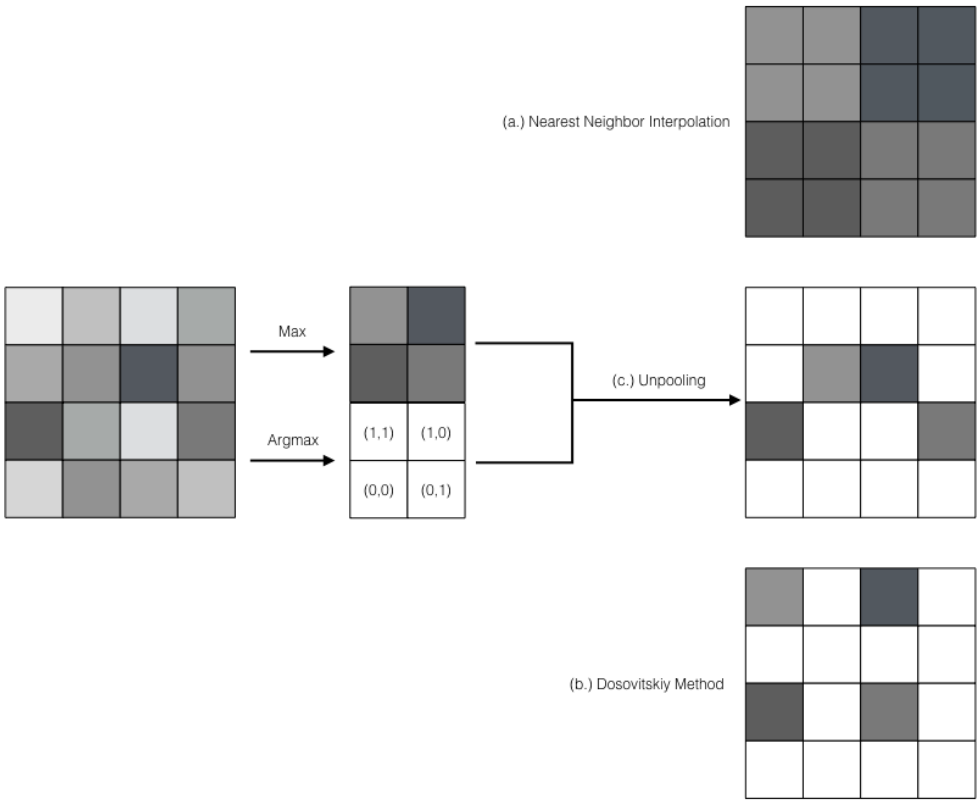

Figure 1. Conceptual illustration of three approaches to inverting the Maxpool operation. (a.) Nearest Neighbor interpolation produces a dense output that loses local structural information from the input. (b.) Dosovitskiy's method [12] produces a sparse output without additional information, however as in interpolation methods informative local structure is lost. (c.) Unpooling as suggested by Zeiler [14] sparse reconstructions of the input that preserve local structure through the addition of argmax or *switch* variables.

as the input to the corresponding max pool. Transpose convolution is a means of densifying, or more explicitly, interpolating this sparse representation with a convolutional filter. As in inverting pooling, a naive solution to accomplish this might be something like bilinear interpolation, where each element of the upsampled output is computed as a linear combination of its neighbors. Unlike pooling, convolutional layers contain learnable parameters of the model. Therefore, it would be desirable if one could incorporate these weights into the interpolation, and further if the interpolation were (optionally) non-linear.

Transpose convolution may also be thought of as **fractionally strided convolution**. In a more general sense, consider the case of upsampling by some factor n this is equivalent to convolution with a fractional input stride of $\frac{1}{n}$ [15]. As illustrated in [6], given a convolution of some kernel of size k with stride $s \geq 0$, a transpose convolution may be equivalently be accomplished by padding each output element with $s - 1$ zeros and performing a convolution with a filter of size k.

Finally, we consider transpose convolution in terms of the justification for its namesake, as a **matrix transpose**, or more specifically, the transpose of a convolution represented as a matrix operation. In the simple case of an input size $n = 3$, kernel size $k = 2$, $s = 1$, $p = 0$. We flatten the input into a vector $v = vec(x) \in \mathbb{R}^{n \cdot n}$ and arrange a Toeplitz matrix with constant non-zero entries along its descending diagonal corresponding to the w_{ij} components of the filter:

$$W = \begin{bmatrix} w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 & 0 & 0 & 0 \\ 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 \\ 0 & 0 & 0 & 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} \end{bmatrix}. \tag{1}$$

Now convolution with the desired kernel becomes a simple matrix multiplication $v \cdot W$, which produces a vector $v' \in \mathbb{R}^4$. Reshaping v' into the desired 2×2 yields the convolved output. Now the explicit relationship between fully connected and convolutional autoencoders becomes obvious, transpose convolution is in fact W^T. Further, as observed in [6], the transpose convolution is the gradient of the corresponding convolution (satisfying certain conditions) for its input. In many modern machine learning libraries, the "forward pass" of a convolutional layer is computed using W and the "backward pass" (used for back propagation) is computed by W^T. In practice, a transpose convolution may be implemented by simply swapping the forward and backward pass.

2. METHODS

2.1 DATA

The test and training dataset consists of synthetically generated radio signals clean of outside interference. We used the GNU Radio [16] software-defined radio (SDR) framework to construct the modulations that generated this data.

A binary file, produced by randomly choosing byte values $[0, 255]$, is the waveforms' input. This binary data is modulated as in-phase and quadrature (I/Q) samples using each of six methods: on-off keying (OOK), Gaussian frequency-shift keying (GFSK), Gaussian minimum-shift keying (GMSK), differential binary phase-shift keying (DBPSK), differential quadrature phase-shift keying (DQPSK), and orthogonal frequency-division multiplexing (OFDM).

For each modulation, the samples are sent to a Nuand™ BladeRF™ software-defined radio (SDR), where they are upconverted to the carrier frequency. The SDR is configured in RF loop-back mode, such that the RF signal is sent and received only within the device's circuitry, and not to an external antenna. This arrangement provides added realism by incorporating the upconversion and radio effects, but without unwanted third-party signals that could pollute the controlled testing.

The signal sampling rate is set so that the number of samples per symbol (N_{SpS}) is consistent for every modulation type, except for OFDM. In contrast with the other modulation techniques, OFDM encodes data on multiple carrier frequencies simultaneously, within the same symbol, and modulates each carrier frequency independently. Our experiment used an existing OFDM signal processing component that operates with a symbol rate different than the other configurations, but with the same sample rate. This rate is identical for both the transmission and reception of the signal. The received RF signal is down-converted at the radio and the resulting I/Q samples are stored for analysis.

The data files need to be arranged into a format and structure for use by our neural network. The I/Q data are split into segments consisting of N_{SpV} samples, or samples per vector. A segment is composed of interleaved I/Q values for each sample, forming a vector of length $2 \times N_{SpV}$. Thus, each vector contains $\frac{N_{SpV}}{N_{SpS}}$ symbols. These vectors are placed into two sets, *train* and *test* (sizes N_{Vtrain} and N_{Vtest}), such that both the modulation type and positions within the set are random. The parameter N_{SpV} is identical for each modulation type for all experiments described in this report. The specific values of all parameters are shown in Table 1.

Table 1. Data generation parameters.

Description	Parameter	Value
Samples per symbol	N_{SpS}	10
Samples per vector	N_{SpV}	225
Number of training vectors	N_{Vtrain}	60000
Number of training vectors per modulation	N_{Vmod}	10000
Number of test vectors	N_{Vtest}	10000

2.2 ADDITION OF NOISE

To assess the performance of our system with a more realistic channel model, we altered the test data set with additive white Gaussian noise (AWGN). AWGN was added to each set of signal modu-

lation types such that for each set the resulting signal-to-noise ratio (SNR) matched a given value. For each of these signal modulation sets, $\{S_{mod}\}$, the added noise power, P_{noise} is:

$$P_{noise(mod)} = \beta \cdot \frac{1}{N_{s(mod)}} \sum_{\{S_{mod}\}} \frac{1}{\tau} \sum_{t=1}^{\tau} [s_t]^2, \tag{2}$$

where $N_{s(mod)}$ is the number of sample vectors for a particular modulation, s_t is an individual signal sample vector of length τ, and β is a factor chosen such that $10\log(P_{\{S\}}/P_{noise})$ matches the desired SNR. Note that all modulations exhibited similar transmitted power, with the exception of OOK, which was slightly larger.

2.3 SEMI-SUPERVISED LEARNING

In [11], it was illustrated that a convolutional autoencoder may be used to accomplish online semi-supervised learning by casting the problem as multi-task leaning. This paradigm is motivated by the observation that a fully unsupervised autoencoder may be used to perform a non-linear dimensionality reduction in the encoder. If a classifier is trained on the features learned by ϕ, this provides a means of incorporating unlabeled data into a classification task. Further, both these objectives may be optimized jointly to perform semi-supervised learning in an online setting. This particular model is referred to as a Stacked What-Where Autoencoder (SWWAE).

In an SWWAE, the encoder is composed of convolutions and max-pooling operations, and the decoder is composed of unpooling operations and convolutions without tied weights. The model is trained to jointly reconstruct its input using an $l2$ loss and classify using negative log-likelihood. To enforce pre-image estimation at each layer of the model, an $l2$ norm is added to the loss function between the input to each encoder convolution and its reconstruction in the decoder.

The method we propose is similar though distinct from [11] in that our model uses a tied weight convolutional transpose operation in the decoder. Advantages of this method are that it requires half the number of trainable parameters of a SWWAE; this is motivated by a desire to reduce the computational complexity of the model and, ideally, regularize the model to reduce the degree of overfitting during training.

2.4 VISUALIZING DISCRIMINATIVE LOCALIZATION

Feature representations learned by deep neural networks are often difficult to analyze and notoriously challenging to debug. The low-level representations that commonly appear in vision tasks have been broadly documented and are relatively well-understood. For instance, CNNs have been shown to learn color patch and edge-detecting filters in early layers of the network topology [14]. Recently, methods for visualization and interpretation of higher level features have been proposed and applied successfully in computer vision [17]. These methods demonstrate that the latent concepts learned by popular models such as AlexNet [18] and VGG16 [19] include filters sensitive to texture patches, global geometric structure, and other low-frequency spatial information.

Increasingly, methods originally developed for computer vision problems have been adapted to the analysis of time domain radio signals [4] [20]. It is inevitable that the advancement of RF spectrum neural networks will be faced with similar model interpretation challenges to those experienced in

computer vision. Therefore, we argue that the application of the feature visualization methods pioneered in the context of vision tasks may be similarly applied to the related challenge of feature learning and interpretation in digital signal processing.

One such visualization is a class activation map (CAM) that produces a class-conditional heatmap over the network's input. CAMs are a recent technique [1] for obtaining class-conditional saliency maps from a convolutional neural network (CNN). A CAM indicates the discriminative regions of an input signal the CNN uses to identify a particular category. Our method utilizes a global average pooling (GAP) layer that outputs a vector of sample-wise average[5] activations for each feature; the alternative, and conventional method for mapping features to categories is to flatten[6] to a feature vector and learn a dense mapping from this arbitrary feature arrangement to categories using a softmax classifier. An undesirable property of the feature flattening method is that the weight matrix mapping features to classes is a black-box, in that the arrangement of its entries is not meaningful. This arbitrary arrangement makes the correlation between a particular feature and a class label difficult to discern directly. GAP enables an interpretable alternative with minimal performance cost and the added benefit of a sample-wise regularization which, we have observed, eliminates the need for dropout to prevent overfitting in most cases.

GAP outputs a vector whose elements are the sample-wise mean for each filter. Given a feature map $f_k(\cdot)$, which produces some output with k features and s samples, GAP simply averages across the sample dimension $g_k = \frac{1}{s}\sum_t^s f_k(x_t)$ for some input x for each of the k features. We then learn a single linear transform ω with shape $k \times c$, where c is the number of target categories. Notice that the rows of of this weight matrix correspond to the average response of the feature set for a particular class.

To obtain a class activation map M_c for a particular class c, we may simply take a weighted feature-wise sum across the output channels of the layer preceding the GAP with each feature weighted by the corresponding ω_{ck}:

$$M_c(x) = \sum_k \omega_{ck} f_k(x). \tag{3}$$

In essence, GAP aggregates its output sample-wise by averaging, and CAM aggregates feature-wise by summing across features weighted by ω. If we repeat this procedure for each class, and then normalize by the max of the CAM, we obtain a heat-map of size $c \times s$. The CAM may then be up-sampled to the original input sample size to have obtain an approximate one-to-one correspondence between category salience of each sample in the input. This method has been similarly applied in images with minimal variation from the algorithm we describe [1].

Optionally, one may also use the CAM method on some arbitrary feature set, that is, without training an entire model with a GAP-based classifier. In our procedure, we trained a CNN in the usual way using a dense mapping from a flattened feature vector. We then froze the weights [7] of the learned features and replaced the classifier with a GAP and softmax and learned this mapping.

[5] 2-D spatial average in the case of images

[6] Concatenate all activations into a single vector

[7] Weight freezing refers to a method of prevent the weight update step of the backpropigation algorithm, usually for a subset of model components to allow fine-tuning of some other part.

3. RESULTS

In general, this work adheres to the experimental protocol for the AMC task proposed in [4], which serves as the foundation for our inquiry; this paradigm is implemented as a mapping of a sequence of in-phase quadrature (IQ) measurements generated by a software-defined radio to a probability distribution over modulation classes.

3.1 TRAINING SNR EVALUATION

Training CNNs on RF data raises the unique question of determining an optimal training SNR, that is, we would like our models to generalize to out of sample noise power, which may or may not be the SNR on which we trained. To study this question, we trained a collection of CNNs on varying SNRs $\{-5, 0, 5, 10, 20\}$ dB.

Our results illustrate that low-positive SNRs yield the best performance under AWGN. In the rest of our experiments we set the training SNR for all modulations to 5 dB, as models trained at this noise level have been observed to provide the best classification performance on this dataset.

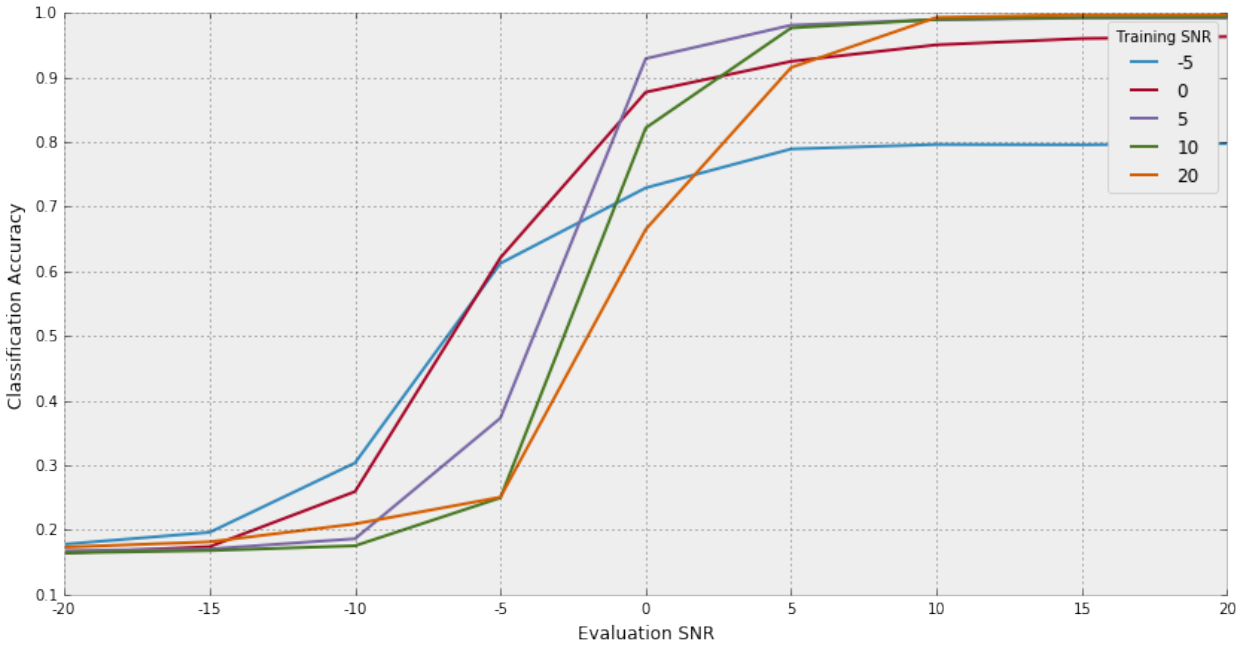

Figure 2. Influence of training noise on model generalization to varying evaluation SNR.

3.2 SEMI-SUPERVISED LEARNING

Unsurprisingly, the predominant effect in these experiments was the quantity of labeled training data. However, we did not observe a clear and consistent trend in the performance of models trained with varying amounts of labeled data as we had predicted. We hypothesize that this may be due to high sensitivity to model initialization on this dataset. Further, we conjecture that it may be possible for multiple objectives to "compete" or dominate one another in a multi-task learning setting. This is

motivated by our observation that the reconstruction loss consistently converges smoothly to an optimum, whereas the classification loss exhibits higher variance during training when unlabeled exam-ples are incorporated in the training set.

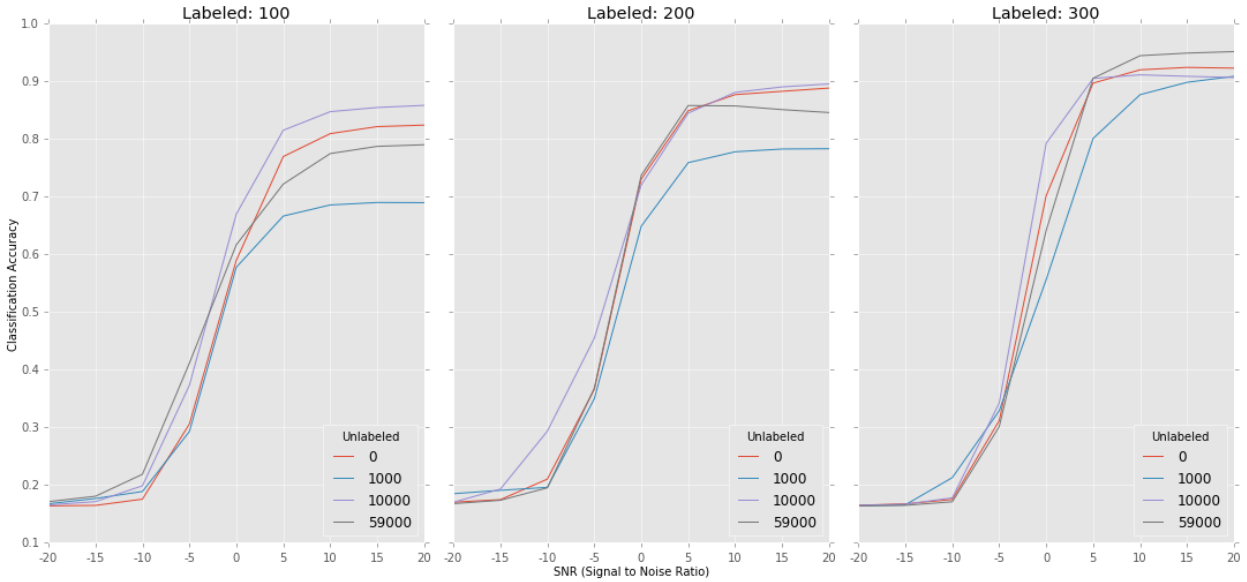

Figure 3. Example results of incorporating variable ratios of labeled and unlabeled examples into the semi-supervised training of a CAE

3.3 CLASS ACTIVATION MAPS

In the following experiments we adopt a protocol similar to [1] in which the authors used several popular pre-trained models, which they adapted to produce CAMs by replacing the fully connected layer with a GAP. Because there are not yet any off-the-shelf models for modulation classification, we simply trained a normal CNN and used the aforementioned procedure to transfer to a GAP layer. We also trained several randomly initialized models that utilized a GAP layer and noticed no significant difference in performance or in the CAMs produced by the two approaches.

In the following sections we will use the term *significance* to refer to learned association of a particular feature to a class. Recall that this is a well-defined quantity ω_{ck}, learned by the weight matrix mapping the sample-wise average of feature k to a class c. Observe the relative response of the network to the correct modulation class as well as the non-target classes. In these diagrams, the two-dimensional I/Q is shown above with the corresponding CAMs for each class below. Each CAM is a time-domain heatmap showing the class-significance weighted sum of feature activations in re-sponse to the input.

The CAMs for on-off keying (OOK) modulated signals show a clear response to the characteristic square waveform of the target class. The CAMs also show positive response on the same interval to classes whose signals possess similar plateau structure in the temporal domain. Note that the sharp transitions on the edges of each symbol are not perceived as strong discriminators for this class, rather they produce low-positive response for OFDM and only show strong significance to the OOK class if the signal is held relatively constant for a relatively longer interval. Conversely, signals that are char-

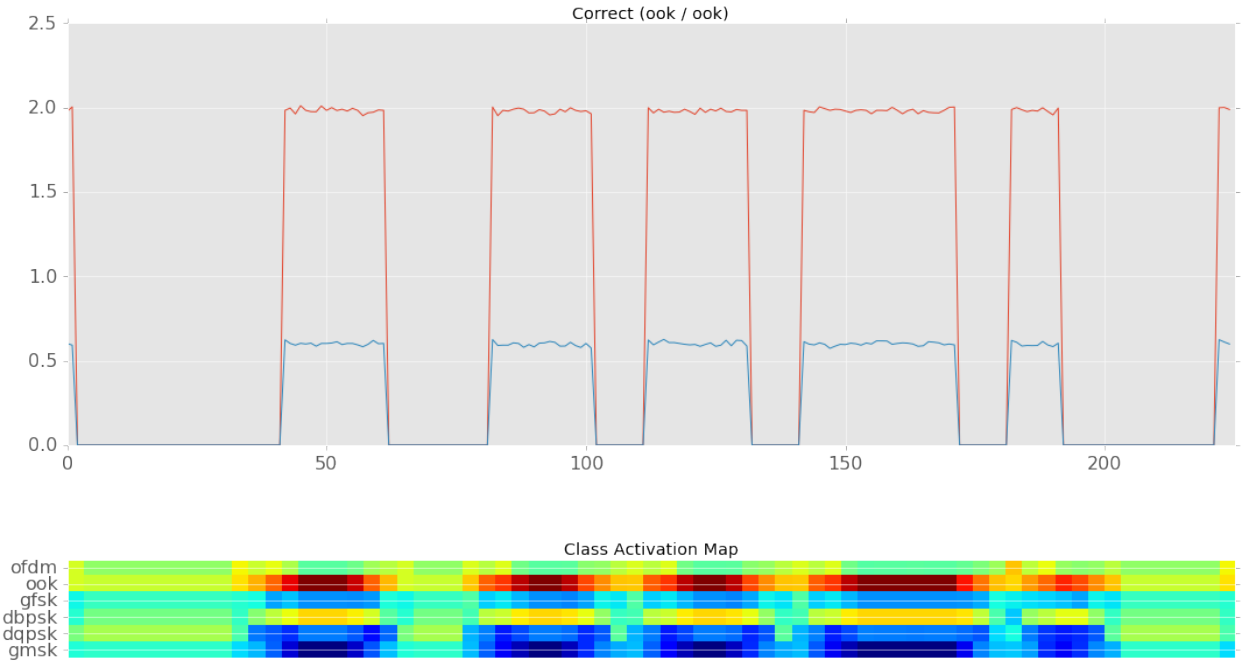

Figure 4. Class activation maps for an OOK modulated signal.

acterized by high-frequency zero crossings (such as GMSK and GFSK) produce low or negative response in the feature mapping to the OOK class.

In the case of Gaussian frequency shift keying (GFSK), the CAM for the target class exhibits a strong sustained response in approximately the first 100 samples of the signal. Note the significance of this structure arises from the particular sustained frequency for that symbol; noting that other modulation types which contain higher frequencies with a similar amplitude in both channels (such as GMSK) do not produce high activations on this interval in the corresponding CAM. Similarly, modulations which are dominated by lower frequencies (such as DQPSK) produce negative values of almost equal magnitude to the GFSK CAM.

Of equivalent value to illustrating where CNNs are successfully learning the class-significant features of data are interpreting the failure cases of such models. The CAM method we propose is well suited for this type of model debugging as well. For instance, a surprising result of this study is that in the case of GFSK, the network assigns much lower significance to samples corresponding to one symbol versus another. Because the data consist of randomly generated binary sequences, where both symbols are equally likely, this cannot be a reflection of learning the class conditional probability one symbol versus another. A more likely explanation is that in the representation the network has learned, one of the symbols is closer than the other to examples of non-target classes in feature space. Slightly more explicitly, consider a case were we subdivide a particular GFSK example $g \in G \subset X$, and produce new examples $\hat{g}_0, \hat{g}_1$ grouped by symbol and project them into the feature space our CNN has leaned. Given a clustering of non-GFSK data with means $\mu_0 ... \mu_{c-1}$, some appropriate distance metric d and some small positive ϵ we hypothesize that $\frac{1}{c-1}\sum_{i=0}^{c-1} d(\hat{g}_0, \mu_i) + \epsilon < \frac{1}{c-1}\sum_{i=0}^{c-1} d(\hat{g}_1, \mu_i)$ for all GFSK examples G. Though we do not conduct an inquiry into this claim here, it is important to note that this analysis could not be intuited without discriminative localization.

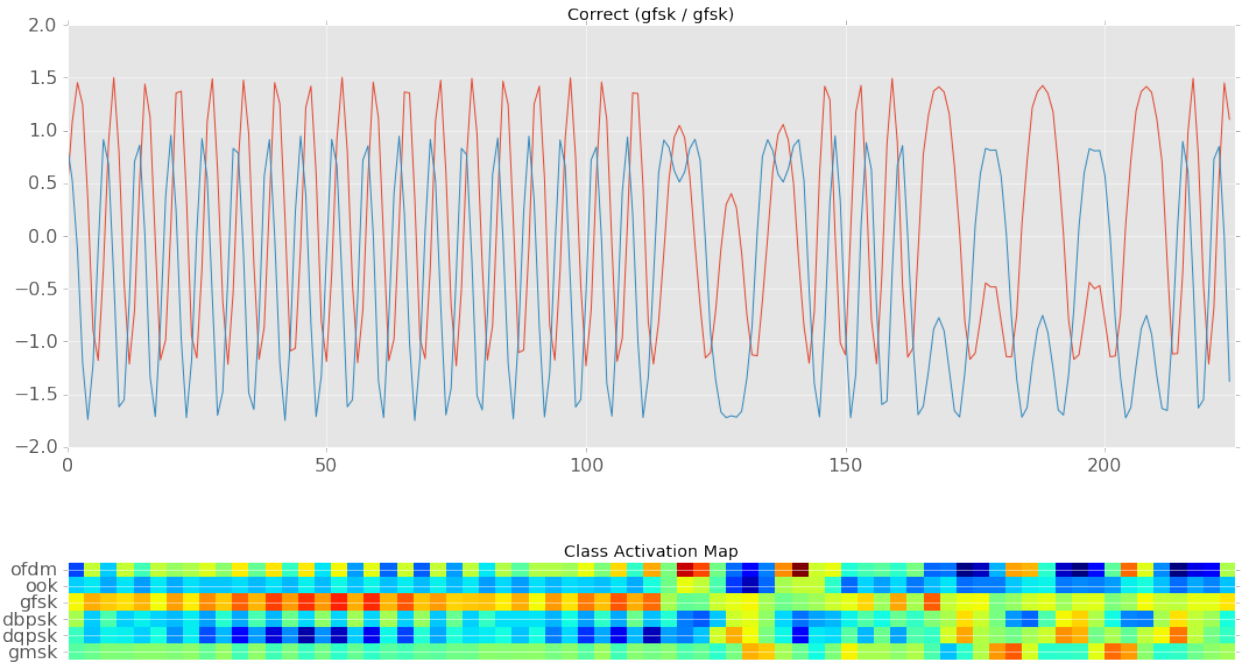

Figure 5. Class activation maps for a GFSK modulated signal.

In Figure 6, we illustrate the impact of noise on discriminative localization using class activation maps. It has been observed, originally in [4] and consistent with our experiments, that neural network automated modulation classification (AMC) models tend to over-predict GMSK and OFDM at low SNR at the expense of other categories. Intuitively, AWGN introduces deviations from the underlying signal that may lead its statistics to match more closely to those of GMSK. If our model has learned features that are sensitive to these characteristics, segments of signals that have similar statistics should show high activation for these two classes. In this example we observe that although the model correctly predicts the DQPSK target class, the CAMs for OFDM, OOK, and GMSK are all highly active. For OOK we observe that a positive plateau in the signal leads to high activation for the corresponding CAM and samples below zero in both channels lead to negative values as this has not been observed in the training data for OOK. Additionally, there are shear transitions between the segment of the signal with high significance for GMSK versus OFDM, suggesting as before a sensitive geometric relationship between the way the model represents OFDM and GMSK.

A final valuable attribute of our CAM method is that the secondary fine tuning of the GAP-based classifier has little to no impact on overall classification performance. Perfect classification accuracy on the held-out test set may be obtained with a standard CNN when no noise is added to the signal. If we then replace the softmax classifier with a GAP layer in the method previously described, after training, only a few examples are misclassified and the model retains 99.9% accuracy. Further, a randomly initialized model with a GAP classifier obtains equivalent performance to a conventional CNN if features are learned jointly with classification.

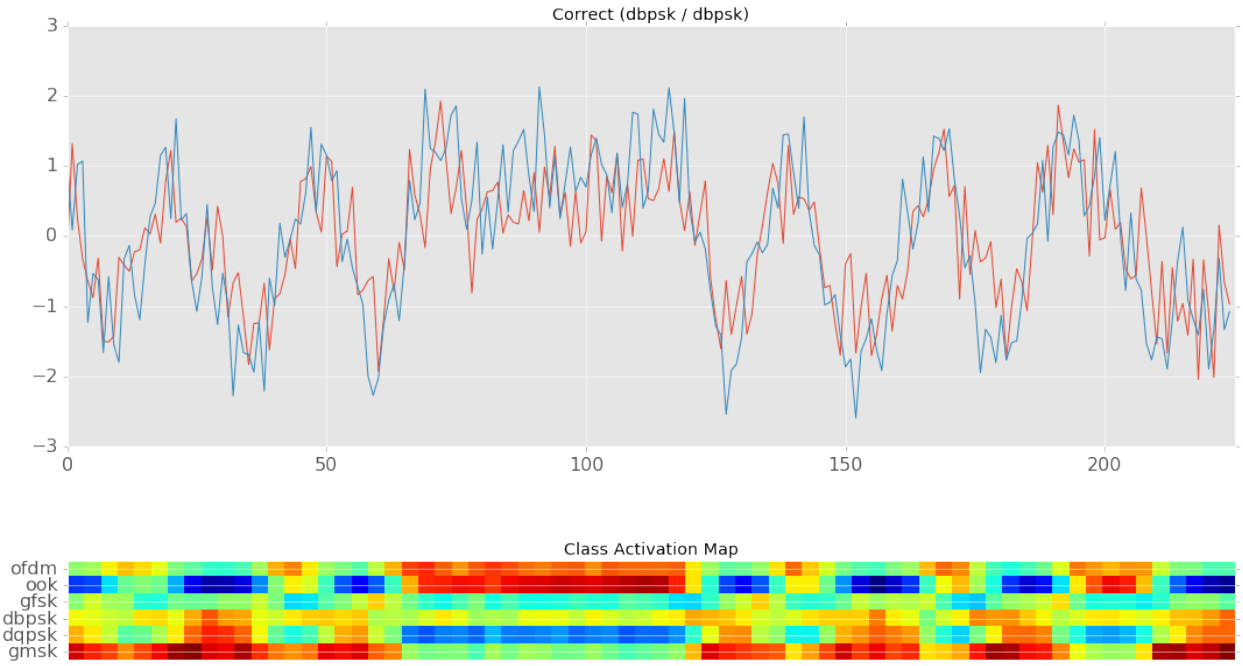

Figure 6. Class activation maps for a DBPSK modulated signal with 5-dB SNR.

4. CONCLUSION

Although this work is by no means the first to define an technique for inverting convolutional neural networks using unpooling and transpose convolutions, we feel that there is not, to the best of our knowledge, an authoritative and approachable explanation of the method. In our background sections we sought to deconflict the oft confusing terminology surrounding transpose convolution and summarize the equivalence of several algorithms for its computation that have been independently proved elsewhere. Our aim has been to provide a single document that would allow the implementation of the most challenging parts of a convolutional autoencoder without the need for an extensive literature review. Further, we have laid the groundwork for future research applying CAEs to the RF spectrum for unsupervised feature learning and denoising. Finally, we have offered a solution to a key limitation of CNN feature interpretation in the RF domain by demonstrating a novel application of class activation maps.

REFERENCES

1. B. Zhou, A. Khosla. A. Lapedriza, A. Oliva, and A. Torralba. 2015. “Learning Deep Features for Discriminative Localization.” Cornell University Library. Computing Research Repository (CoRR). Available online at http://arxiv.org/abs/1512.04150. Accessed September 15, 2016.

2. P. Baldi, and K. Hornik. 1989. “Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima,” Neural Network, vol. 2, no. 1, pp. 53–58 (January). Availalable online at http://dx.doi.org/10.1016/0893-6080(89)90014-2. Accessed September 15, 2016.

3. Y. Bengio. 2009. “Learning Deep Architectures for AI (Foundations and Trends in Machine Learning), vol. 2, no. 1, pp. 1–127. Available online at http://dx.doi.org/10.1561/2200000006. Accessed Spetember 15, 2016.

4. B. Migliori, R. Zeller-Townson, D. Grady, and D. Gebhardt. 2016. “Biologically Inspired Radio Signal Feature Extraction with Sparse Denoising Autoencoders.” Technical Report 3047. Space and Naval Warfare Systems Center Pacific (SSC Pacific). San Diego, CA.

5. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958. Available online at http://jmlr.org/papers/v15/srivastava14a.html. Accessed September 20,2016.

6. V. Dumoulin, and F. Visin. 2016. “A Guide to Convolution Arithmetic for Deep Learning.” Cornell University Library. Computing Research Repository (CoRR). Available online at https://arxiv.org/abs/1603.07285. Accessed September 15, 2016.

7. M. D. Zeiler, and R. Fergus. 2013. “Visualizing and Understanding Convolutional Networks.” Cornell University Library. Computing Research Repository (CoRR). Available online at http://arxiv.org/abs/1311. Accessed September 15, 2016.

8. V. Badrinarayanan, A. Kendall, and R. Cipolla. 2015. “Segnet: A Deep Convolutional Encoder-decoder Architecture for Image Segmentation.” Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1511.00561. Available online at http://arxiv.org/abs/1511.00561. Accessed September 15, 2016.

9. A. Radford, L. Metz, and S. Chintala. 2015. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1511.06434. Available online at http://arxiv.org/abs/1511.06434/. Accessed September 15, 2016.

10. D. P. Kingma, and M. Welling. 2013. “Auto-encoding Variation Bayes.”

11. J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, “Stacked what-where auto-encoders,” 2015. http://arXiv.org/abs/1603.07285. Accessed September 15, 2016.

12. A. Dosovitskiy, J. T. Springenberg, and T. Brox. 2014. “Learning to Generate Chairs with Convolutional Neural Networks,” Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1411.5928. Available onlinea at http://arxiv.org/abs/1411.5928. Accessed September 15, 2016

13. A. Dosovitskiy, and T. Brox. 2015. "Inverting Convolutional Networks with Convolutional Networks." Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1506.02753. Available online at http://arxiv.org/abs/1506.02753. Accessed September 15, 2016.

14. M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. 2010. "Deconvolutional Networks." Computer Vision Papers (CVPR). Available online at http://www.matthewzeiler.com/pubs/cvpr2010/cvpr2010.pdf. Accessed September 15, 2016.

15. J. Long, E. Shelhamer, and T. Darrell. 2014. "Fully Convolutional Networks for Semantic Segmentation." Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1411.4038. Available at http://arxiv.org/abs/1411.4038. Accessed September 15, 2016.

16. E. Blossom. 2004. "Gnu Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, vol. 2004, no. 122 (June), p. 4. Available online at http://dl.acm.org/citation.cfm?id=993247.99325115. Accessed September 15, 2016.

17. A. Mahendran, and A. Vedaldi. 2014. "Understanding Deep Image Representations by Inverting Them." Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1412.0035, 2014. Available online at http://arxiv.org/abs/1412.0035. Accessed September 15, 2016.

18. A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. "Imagenet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems, Volume 25* (pp. 1097–1105). F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., Red Hook, NY. Available online at http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Accessed September 15, 2016.

19. K. Simonyan, and A. Zisserman. 2014. "Very Deep Convolutional Networks for Large-Scale Image Recognition. " Computing Research Repository (CoRR), Cornell University Library. CoRR, vol. abs/1409.1556.

20. T. J. O'Shea, J. Corgan, and T. C. Clancy. 2016. "Convolutional Radio Modulation Recognition Networks." Available online at https://arxiv.org/abs/1602.04105. Accessed September 15, 2016.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
September 2016	Final	

4. TITLE AND SUBTITLE

Learning and Visualizing Modulation Discriminative Radio Signal Features

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Michael Walton
Daniel Gebhardt
Benjamin Migliori
Logan Straatemeier

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SSC Pacific
53560 Hull Street
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 3048

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

SSC Pacific
Naval Innovative Science and Engineering (NISE) Program
53560 Hull Street
San Diego, CA 92152–5001

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this work we explore the adaptation of convolutional autoencoders to complex-valued temporal domain radio signals. We propose a method for accomplishing online semi-supervised learning with a tied-weight convolutional autoencoder applied to a modulation classification task and provide some initial results. We also demonstrate a novel application of class activation maps (CAMs) to obtain interpretable visualizations of modulation-discriminative temporal structure in input signals. Finally, we show that our visualization method may be successfully applied to pre-trained models with negligible impact on classification performance on an automated modulation classification (AMC) task. This work was done as part of the BIAS (Biologically Inspired Autonomous Sensing) project, funded from the Naval Innovative Science and Engineering (NISE) Program.

15. SUBJECT TERMS

tied-weight convolutional autoencoders; complex-valued temporal domain radio signals; online semi-supervised learning; class activation maps; modulation-discriminative temporal structure; automated modulation classification

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Michael Walton
U	U	U	U	24	**19B. TELEPHONE NUMBER** *(Include area code)* 619-553-2421

INITIAL DISTRIBUTION

84300	Library	(1)
85300	Archive/Stock	(1)
56120	D. Gebhardt	(1)
56140	L. Straatemeir	(1)
56150	B. Migliori	(1)
56150	M. Walton	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

Approved for public release.

SSC Pacific
San Diego, CA 92152-5001